FIG. 1

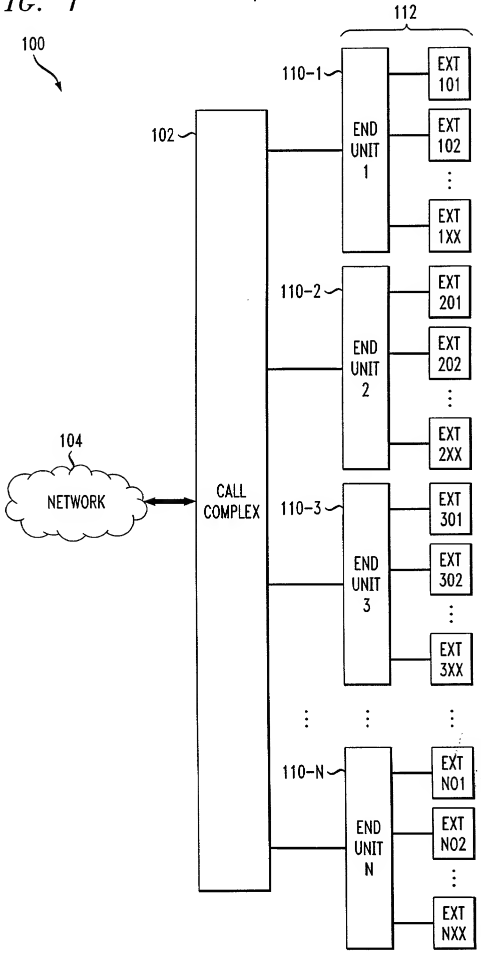

100
102
104
NETWORK
CALL COMPLEX
112
110-1
110-2
110-3
110-N
END UNIT 1
END UNIT 2
END UNIT 3
END UNIT N
EXT 101
EXT 102
EXT 1XX
EXT 201
EXT 202
EXT 2XX
EXT 301
EXT 302
EXT 3XX
EXT N01
EXT N02
EXT NXX

FIG. 2

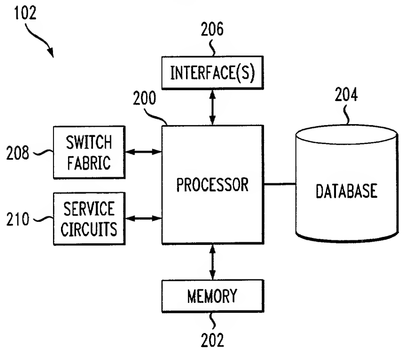

102
206
INTERFACE(S)
200
204
208
SWITCH FABRIC
PROCESSOR
DATABASE
210
SERVICE CIRCUITS
MEMORY
202

FIG. 3

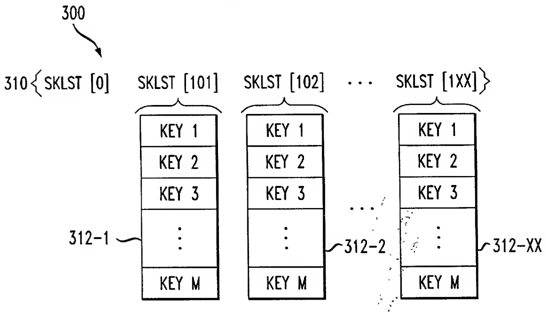

300
310
SKLST [0]
SKLST [101]
SKLST [102]
...
SKLST [1XX]
KEY 1
KEY 2
KEY 3
KEY M
312-1
312-2
312-XX

FIG. 4

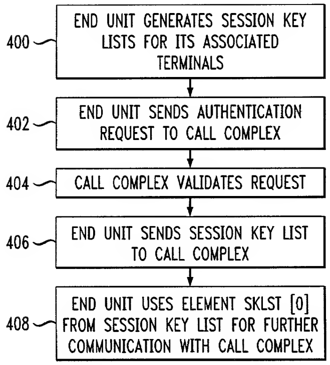

400
END UNIT GENERATES SESSION KEY LISTS FOR ITS ASSOCIATED TERMINALS
402
END UNIT SENDS AUTHENTICATION REQUEST TO CALL COMPLEX
404
CALL COMPLEX VALIDATES REQUEST
406
END UNIT SENDS SESSION KEY LIST TO CALL COMPLEX
408
END UNIT USES ELEMENT SKLST [0] FROM SESSION KEY LIST FOR FURTHER COMMUNICATION WITH CALL COMPLEX

FIG. 5

CALL COMPLEX		END UNIT

	←	END UNIT$_{\text{SESSION KEY}}$ = RANDOM()
		ESKe = ENCRYPT (END UNIT$_{\text{SESSION KEY}}$) END UNIT$_{\text{PRIVATE KEY}}$
		EEUIDe = ENCRYPT (END UNIT IDENTIFICATION)
		CALL COMPLEX$_{\text{PUBLIC KEY}}$
		SendAuthenticationReq (EEUIDe, ESKe)

IDENTIFY REQUEST (VALIDATE REQUEST; IF IT IS NOT VALID, DROP IT)
END UNIT IDENTIFICATION = DECRYPT (EEUIDe)
CALL COMPLEX$_{\text{PRIVATE KEY}}$
IF (END UNIT IDENTIFICATION) EXISTS GET END UNIT$_{\text{PUBLIC KEY}}$
END UNIT$_{\text{SESSION KEY}}$ = ENCRYPT (END UNIT$_{\text{SESSION KEY}}$)
END UNIT$_{\text{PUBLIC KEY}}$
ACKe = ENCRYPT (ACK) END UNIT$_{\text{SESSION KEY}}$
CreateSessionInformation (IP ADDRESS, END UNIT IDENTIFICATION)
SendRegistrationAcknowledgement (ACKe)
	→	

		SKLSTe = ENCRYPT (GenerateSessionKeyListForEndUnit())
		END UNIT$_{\text{SESSION KEY}}$
		SendSessionKeyList (SKLSTe)

SKLST = DECRYPT (SKLSTe)
END UNIT$_{\text{SESSION KEY}}$ = SKLST[0]
ACKe = ENCRYPT (ACK) END UNIT$_{\text{SESSION KEY}}$
SendSessionKeyListAcknowledgement (ACKe)
		END UNIT$_{\text{SESSION KEY}}$ = SKLST[0]

FIG. 6A

CALL COMPLEX		END UNIT 1
	←	CallRequestTo (EXTENSION 201, EXTENSION 105) END UNIT$_{\text{SESSION KEY}}$

IF INCOMING REQUEST IP ADDRESS NOT REGISTERED, DROP THE REQUEST
END UNIT$_{\text{SESSION KEY}}$ = FIND SESSION KEY FOR IP (REQUEST IP ADDRESS)
CALL REQUEST DATA = DECRYPT (INCOMING BUFFER) END UNIT$_{\text{SESSION KEY}}$
IF PLAINTEXT BUFFER DOES NOT CONTAIN END UNIT REGISTRATION NAME, DROP THE REQUEST		

CALL COMPLEX		**END UNIT 2**
EUEUSK = SKLST[105] MESSAGE KEY = get_key_for_extension (201) SendIncomingCallRequest (ENCRYPT (oIP, 201, 105, EUEUSK)$_{\text{MESSAGE KEY}}$)	→	

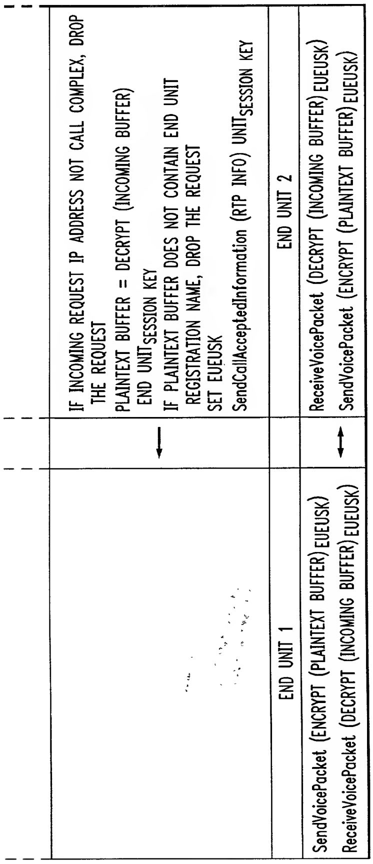

IF INCOMING REQUEST IP ADDRESS NOT CALL COMPLEX, DROP THE REQUEST
PLAINTEXT BUFFER = DECRYPT (INCOMING BUFFER) END UNIT$_{SESSION KEY}$
IF PLAINTEXT BUFFER DOES NOT CONTAIN END UNIT REGISTRATION NAME, DROP THE REQUEST
SET EUEUSK
SendCallAcceptedInformation (RTP INFO) UNIT$_{SESSION KEY}$
END UNIT 1
END UNIT 2
SendVoicePacket (ENCRYPT (PLAINTEXT BUFFER)$_{EUEUSK}$)
ReceiveVoicePacket (DECRYPT (INCOMING BUFFER)$_{EUEUSK}$)
ReceiveVoicePacket (DECRYPT (INCOMING BUFFER)$_{EUEUSK}$)
SendVoicePacket (ENCRYPT (PLAINTEXT BUFFER)$_{EUEUSK}$)

FIG. 6A CONT.

FIG. 6B

END UNIT 1		CALL COMPLEX

ConfRequestTo (EXTENSION 311, EXTENSION 105) END UNIT$_{\text{SESSION KEY}}$	→	IF INCOMING REQUEST IP ADDRESS NOT REGISTERED, DROP THE REQUEST
		END UNIT$_{\text{SESSION KEY}}$ = FIND SESSION KEY FOR IP (REQUEST IP ADDRESS)
		CALL REQUEST DATA = DECRYPT (INCOMING BUFFER) END UNIT$_{\text{SESSION KEY}}$
		IF PLAINTEXT BUFFER DOES NOT CONTAIN END UNIT REGISTRATION NAME, DROP THE REQUEST

END UNIT 3		CALL COMPLEX
	←	EUEUSK = SKLST[105] MESSAGE KEY = get_key_for_extension (311) SendIncomingConfRequest (ENCRYPT (oIP, 511, 105, EUEUSK)$_{\text{MESSAGE KEY}}$)

IF INCOMING REQUEST IP ADDRESS NOT CALL COMPLEX, DROP THE REQUEST
PLAINTEXT BUFFER = DECRYPT (INCOMING BUFFER) END UNIT$_{\text{SESSION KEY}}$
IF PLAINTEXT BUFFER DOES NOT CONTAIN END UNIT REGISTRATION NAME, DROP THE REQUEST
SET EUEUSK
SendConfAcceptedInformation (RTP INFO) UNIT$_{\text{SESSION KEY}}$		

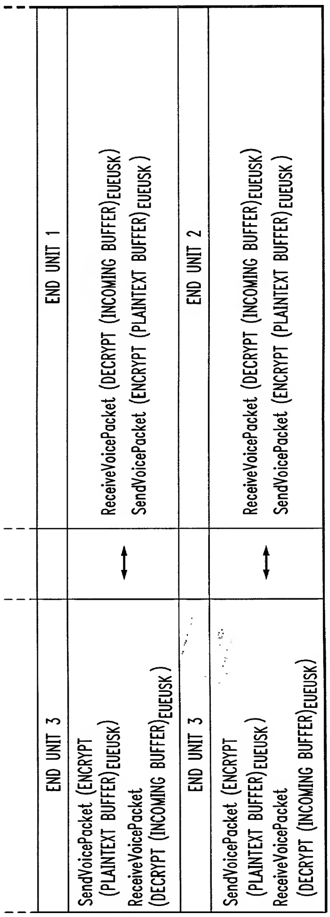

END UNIT 3
SendVoicePacket (ENCRYPT (PLAINTEXT BUFFER)EUEUSK)
ReceiveVoicePacket (DECRYPT (INCOMING BUFFER)EUEUSK)
END UNIT 1
ReceiveVoicePacket (DECRYPT (INCOMING BUFFER)EUEUSK)
SendVoicePacket (ENCRYPT (PLAINTEXT BUFFER)EUEUSK)
END UNIT 3
SendVoicePacket (ENCRYPT (PLAINTEXT BUFFER)EUEUSK)
ReceiveVoicePacket (DECRYPT (INCOMING BUFFER)EUEUSK)
END UNIT 2
ReceiveVoicePacket (DECRYPT (INCOMING BUFFER)EUEUSK)
SendVoicePacket (ENCRYPT (PLAINTEXT BUFFER)EUEUSK)

FIG. 6B CONT.

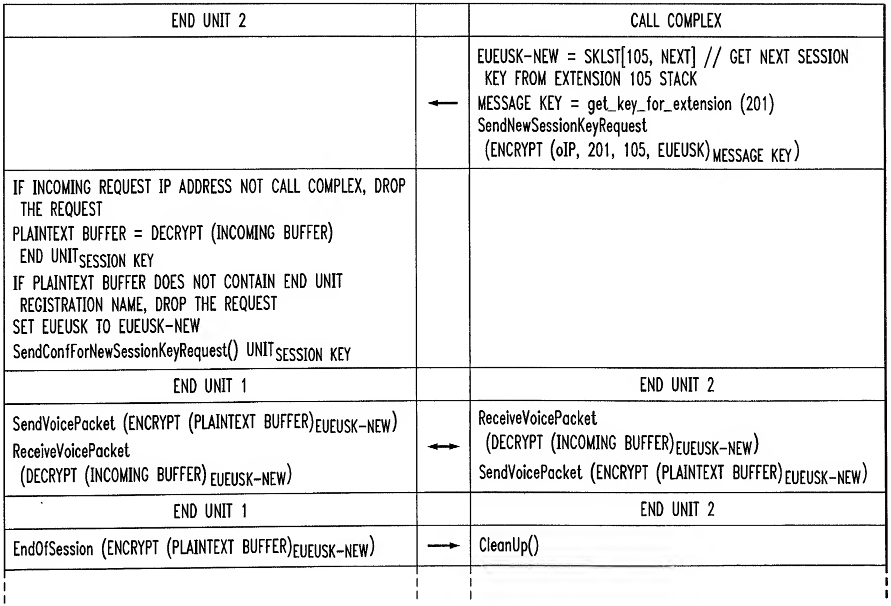
FIG. 6C

END UNIT 1		CALL COMPLEX

DropSession (EXTENSION 311) END UNIT$_{\text{SESSION KEY}}$	→	IF INCOMING REQUEST IP ADDRESS NOT REGISTERED, DROP THE REQUEST
		END UNIT$_{\text{SESSION KEY}}$ = FIND SESSION KEY FOR IP (REQUEST IP ADDRESS)
		CALL REQUEST DATA = DECRYPT (INCOMING BUFFER) END UNIT$_{\text{SESSION KEY}}$
		IF PLAINTEXT BUFFER DOES NOT CONTAIN END UNIT REGISTRATION NAME, DROP THE REQUEST

END UNIT 3		CALL COMPLEX
CleanUp()	←	DropSession (EXTENSION 311) END UNIT$_{\text{SESSION KEY}}$

FIG. 6D

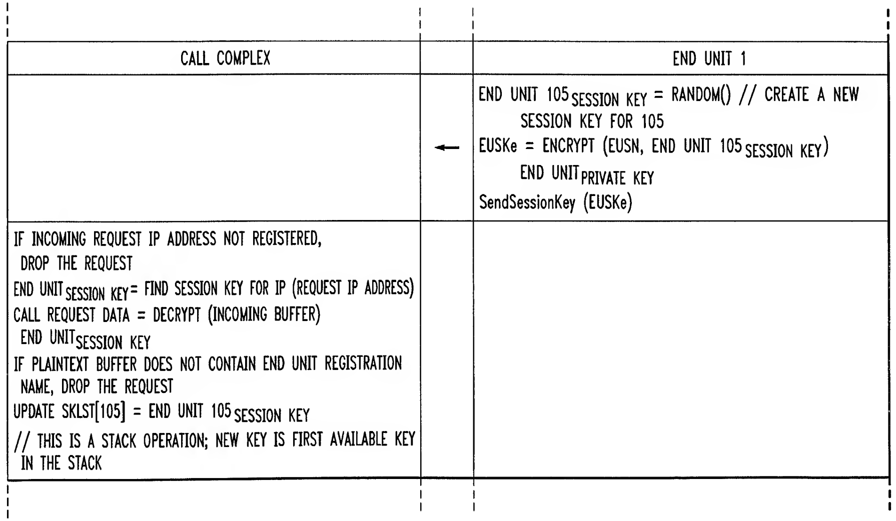


END UNIT 2		CALL COMPLEX
	←	EUEUSK-NEW = SKLST[105, NEXT] // GET NEXT SESSION KEY FROM EXTENSION 105 STACK MESSAGE KEY = get_key_for_extension (201) SendNewSessionKeyRequest (ENCRYPT (oIP, 201, 105, EUEUSK)$_{\text{MESSAGE KEY}}$)

IF INCOMING REQUEST IP ADDRESS NOT CALL COMPLEX, DROP THE REQUEST
PLAINTEXT BUFFER = DECRYPT (INCOMING BUFFER) END UNIT$_{\text{SESSION KEY}}$
IF PLAINTEXT BUFFER DOES NOT CONTAIN END UNIT REGISTRATION NAME, DROP THE REQUEST
SET EUEUSK TO EUEUSK-NEW
SendConfForNewSessionKeyRequest() UNIT$_{\text{SESSION KEY}}$		

END UNIT 1		END UNIT 2
SendVoicePacket (ENCRYPT (PLAINTEXT BUFFER)$_{\text{EUEUSK-NEW}}$) ReceiveVoicePacket (DECRYPT (INCOMING BUFFER)$_{\text{EUEUSK-NEW}}$)	↔	ReceiveVoicePacket (DECRYPT (INCOMING BUFFER)$_{\text{EUEUSK-NEW}}$) SendVoicePacket (ENCRYPT (PLAINTEXT BUFFER)$_{\text{EUEUSK-NEW}}$)
END UNIT 1		END UNIT 2
EndOfSession (ENCRYPT (PLAINTEXT BUFFER)$_{\text{EUEUSK-NEW}}$)	→	CleanUp()

FIG. 6D CONT. (1)

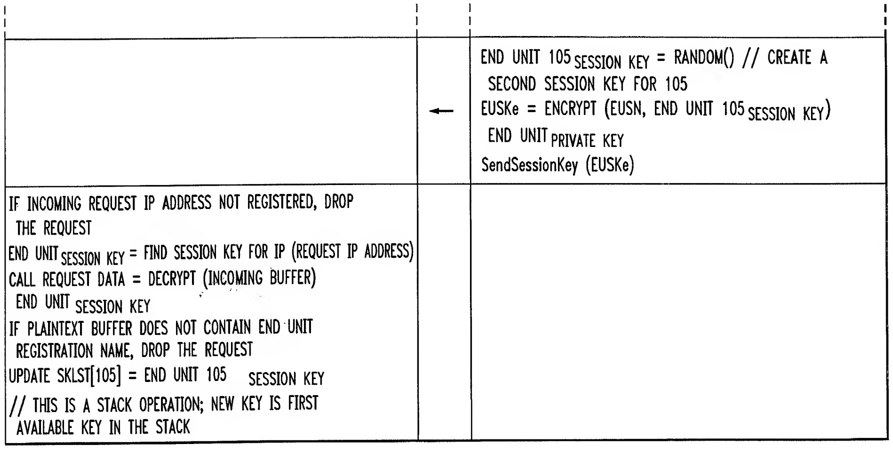


CALL COMPLEX		END UNIT 1

	←	END UNIT 105$_{\text{SESSION KEY}}$ = RANDOM() // CREATE A NEW SESSION KEY FOR 105
		EUSKe = ENCRYPT (EUSN, END UNIT 105$_{\text{SESSION KEY}}$) END UNIT$_{\text{PRIVATE KEY}}$
		SendSessionKey (EUSKe)

IF INCOMING REQUEST IP ADDRESS NOT REGISTERED, DROP THE REQUEST
END UNIT$_{\text{SESSION KEY}}$= FIND SESSION KEY FOR IP (REQUEST IP ADDRESS)
CALL REQUEST DATA = DECRYPT (INCOMING BUFFER) END UNIT$_{\text{SESSION KEY}}$
IF PLAINTEXT BUFFER DOES NOT CONTAIN END UNIT REGISTRATION NAME, DROP THE REQUEST
UPDATE SKLST[105] = END UNIT 105$_{\text{SESSION KEY}}$
// THIS IS A STACK OPERATION; NEW KEY IS FIRST AVAILABLE KEY IN THE STACK		

FIG. 6D CONT. (2)



```
END UNIT 105$_{SESSION\ KEY}$ = RANDOM() // CREATE A
 SECOND SESSION KEY FOR 105
EUSKe = ENCRYPT (EUSN, END UNIT 105$_{SESSION\ KEY}$)
 END UNIT$_{PRIVATE\ KEY}$
SendSessionKey (EUSKe)
```

```
IF INCOMING REQUEST IP ADDRESS NOT REGISTERED, DROP
 THE REQUEST
END UNIT$_{SESSION\ KEY}$ = FIND SESSION KEY FOR IP (REQUEST IP ADDRESS)
CALL REQUEST DATA = DECRYPT (INCOMING BUFFER)
 END UNIT$_{SESSION\ KEY}$
IF PLAINTEXT BUFFER DOES NOT CONTAIN END UNIT
 REGISTRATION NAME, DROP THE REQUEST
UPDATE SKLST[105] = END UNIT 105$_{SESSION\ KEY}$
// THIS IS A STACK OPERATION; NEW KEY IS FIRST
 AVAILABLE KEY IN THE STACK
```